KYBERNETIKA — VOLUME 26 (1990), NUMBER 6

ISARITHMIC FLOW CONTROL USING LEARNING AUTOMATA

A. V. VASILAKOS, A. HARITSIS, S. BATISTATOS

The main objective of flow control in a store-and forward packet switched network is a good tradeoff between throughput and delay. The isarithmic method is an algorithm for network access level flow control [6], that allows packets enter the subnet only if a free "permit" exists at the source-node.

A learning automaton is situated at each exit-node, attempting to make an optimal decision for the distribution of permits. We assume a network with Virtual Circuits (VC) and we analyze the performance of the "Isarithmic-Learning" algorithm.

An Event Driven Simulator has been derived for the comparison of "Isarithmic-Learning" algorithm with "Isarithmic-Random" algorithm (random distribution of permits).

1. INTRODUCTION

The main idea in isarithmic flow is controlling the congestion by limiting the total number of packets existing in the network at any instant of time. This restriction is achieved by having a fixed number of "permits" in the whole network, stored at the nodes and travelling with packets. In order to enter the subnet, a packet must capture one of the permits stored in the source-node, otherwise it waits outside the network. After arriving at the destination, the permit is released by the packet and can be returned to any node.

Problems. The particular problems appeared at isarithmic flow control are listed below:

1) Although the global congestion is prevented, it does not guarantee that there will not be an accumulation of permits anywhere, leading to congestion in that point of the network.
2) It is difficult to find a good algorithm for the permits distribution. If they are returned randomly, it is certain that every node will have some of them and a new packet will not suffer a large delay before capturing a permit and leaving the source-node. This random distribution may cause problems, when the arrival rate packets for a Virtual Circuit (VC) is large (e.g. file transfer) related to another

Virtual Circuit. Then, there will not exist enough permits for the service of the first VC (permit starvation).

3) The disappearance of generation of permits must be avoided.

A Solution in the Permit-Distribution Problem

From the above remarks, it is obvious that isarithmic flow control is not enough for an efficient network operation. A combination of isarithmic – ETE window flow control for each VC is needed [2, 6].

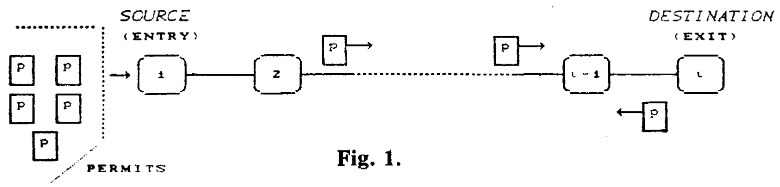

Fig. 1.

Permits are freed at the exit-node. A Learning Automaton is placed there and chooses the VC, among them terminating at the same node, that will obtain the permit. It is the Learning Automaton that undertakes the ETE flow control.

2. BASIC ELEMENTS FOR THE LEARNING AUTOMATON

A Learning Automaton (LA) is defined as a fivetuple $\{\alpha, \beta, \phi, p, T\}$ [7], where

α: the set $\{\alpha_1, \alpha_2, \ldots, \alpha_r\}$ of automaton actions.

β: a continuous variable on $[0, 1]$, the automaton input (S-model).

ϕ: the automaton strategy.

p: the set $\{p_1, p_2, \ldots, p_r\}$ of action probabilities.

T: the learning algorithm.

The environment is defined as a triple $\{\alpha, \beta, c\}$ [7], where

α: the set $\{\alpha_1, \alpha_2, \ldots, \alpha_r\}$ of environment input.

β: a continuous variable on $[0, 1]$, the environment output.

c: the set $\{c_1, c_2, \ldots, c_r\}$ of penalty probabilities

$$c_i(n) = \mathrm{P}[\beta(n) = 1 \mid \alpha(n) = \alpha_i], \quad c_i \in [0, 1].$$

We assume a non-stationary environment, thus the $c_i(n)$ change with time.

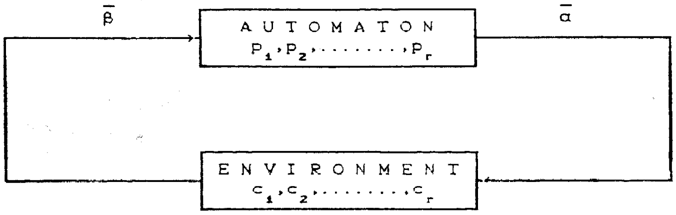

Fig. 2.

The automaton operation is explained below: At instant n, the automaton chooses an action $\alpha(n) = \alpha_i$ with probability $p_i(n)$. The environment responds with a feedback $\beta(n)$, that causes REWARD or PENALTY on the selected action. The action probabilities are updated:

$$p(n+1) = T[p(n), \alpha(n), \beta(n)]$$

2.1 Learning Algorithm

There are linear or non-linear learning algorithms, with the $p(n+1)$ being a linear or non-linear function of $p(n)$ correspondingly. In addition, there are hybrid algorithms that combine the previous ones. In our implementation the linear SL_{RP} is used [7].

If the selected action is $\alpha(n) = \alpha_i$, the following forms for the action probabilities are used [7]:

REWARD	$p_i(n+1) = p_i(n) + a[1 - c_i(n)]\,[1 - p_i(n)]$	
$\beta(n) = 0$	$p_j(n+1) = p_j(n) - a[1 - c_i(n)]\,p_j(n)$	(1)
PENALTY	$p_i(n+1) = p_i(n) - b\,c_i(n)\,p_i(n)$	
$\beta(n) = 1$	$p_j(n+1) = p_j(n) + b\,c_i(n)\left[\dfrac{1}{r-1} - p_j(n)\right]$	

where a, b: parameters on $[0, 1]$

j: $1, 2, \ldots, i-1, i+1, \ldots, r$.

For a non-penalty input (REWARD-case), the p_i is increased and the other probabilities are decreased. On the PENALTY-case, p_i is decreased and all the other components of p are increased. In both cases, the summation of probabilities remains unchanged and equal to 1.

2.2 Classification of Linear Learning Algorithms

Based on the forms (1), we can classify the excited learning algorithms into three categories [3], [4]:

1) Linear Reward-Inaction (L_{RI}): In this case $b = 0$. This algorithm is non-ergodic and ε-optimal. The action probabilities $p_i(n)$ converge to a limited random variable with probability one.

2) Linear Reward-Penalty (L_{RP}): The parameters a, b are equal. The algorithm is ergodic and expedient. The action probabilities converge to a random variable with continuous distribution.

3) Linear ($L_{R\varepsilon P}$): In this case $b = 0[a]$ where $b/a \to 0$ as $a \to 0$. It is ergodic

and with a proper choice of b, the algorithm approaches SL_{RI} and as a result becomes ε-optimal.

In our implementation we have preferred $SL_{R\varepsilon P}$.

2.3 Computation of Penalty Probabilities

We have assumed non-stationary environment for the LA. The penalty probabilities $c_i(n)$ change with the time. There are two different ways of computation [7]:

1. The penalty probabilities depend on the action probabilities:

$$c_i(n+1) = \phi_i[p_i(n)]$$

2. The penalty probabilities are affected from their previous values and the action probabilities:

$$c_i(n+1) = k\, c_i(n) + (1-k)\, \phi_i[p_i(n)]\,, \quad k \in [0, 1] \tag{2}$$

The function $\phi_i[p_i(n)]$ has the properties [7]:

1. It is defined everywhere on [0, 1]
2. It is continuous on every point of [0, 1]
3. It is monotonically increasing. (3)

In our implementation we have selected the second way of computation for the $c_i(n)$.

3. NETWORK MODEL

For the purpose of network analysis, we model each link as an $M/M/1$ queueing system (cf. Fig. 3) with infinite number of buffers. The basic elements of the queue for link 1 are:

- mean packet length b bits
- capacity $s(1)$ bps
- service rate $c(1) = s(1)/b$
- arrival rate $\gamma(1)$, equal to the summation of the throughputs of all the VC's passing through link 1.

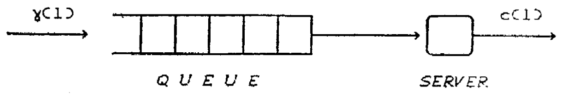

Fig. 3.

The following assumptions are issued for the model:

1. The arrivals at the queue follow Poisson distribution with rate $\gamma(1)$.
2. The packet size follows exponential distribution with average value b.

3. The Kleinrock Independence assumption is issued. Each packet arriving at a node, loses its length and obtains a new length from an exponential distribution with average value *b*.
4. The packet delay includes the Queueing Time and the Transmission Time. So, the mean steady-state packet delay is:

$$\text{DELAY} = \frac{1}{c(1) - \gamma(1)}$$

Speaking about permits, there is an initial fixed number of them at each source-node of every VC. A packet, travelling from the HOST to a source-node, is accepted only if there is at least one permit waiting there. In the case of shortage of permits, the packet is discarded and the HOST retries to send this packet with another EVENT, later, Fig. 4.

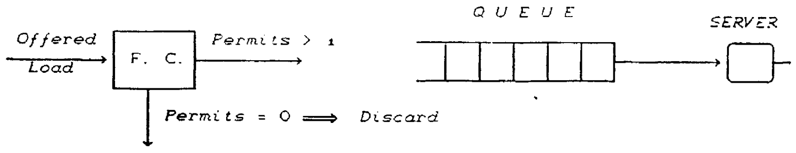

Fig. 4.

Permits travel together with packets (one per packet) until the destination node is reached. At this point, permits are released and the Learning Automaton decides to which VC, having the same destination, they will be sent.

4. THE ISARITHMIC-LEARNING ALGORITHM

Any time that a packet arrives at its destination one permit is released and is directed to the (source) node decided by the LA. Then, the environment of the LA and the state probabilities are updated. The Algorithm's steps follows below:

SOURCE NODE OF *i*th VC	MESSAGE	DESTINATION NODE (LA)
		Selection of action (*i*th VC)
	PERMIT $\longleftarrow$	
Permits := Permits +1		
	$\xrightarrow{T_i}$	
		Updating of penalty probability $c_i(n)$ Updating of state probabilities $p_i(n)$

In order to reduce the execution overhead and to achieve better measures of the environment the Algorithm is executed (and thus the selected action changes), after the arrival of *N* permits at the destination node (INTERVAL = *N*). When the

(kN)th permit arrives $(k = 1, 2, \ldots)$ the Algorithm is executed and its decision remains for the interval of the next N permit arrivals.

4.1 Action Selection

In each LA there is a vector $\boldsymbol{p}$ which represents the action probabilities. Probability p_i expresses the proportion of the permits that the ith VC of this LA will 'consume'. The way an action is selected is as follows:

Probabilities p_i, represented as small line sections (of length relevant to their magnitude), are placed in a row forming a line section of length 1 (see Fig. 5). A random number RND is generated in the range (0, 1). This number is located on the above formed line. If the p_i line section comprises this number, action i (i.e. VC i) is selected for the next N permit arrivals.

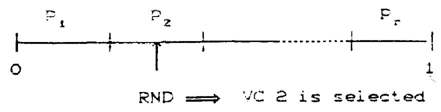

Fig. 5.

4.2 Updating of the Penalty Probabilities

Formula (2) is applied with $\phi_i[p_i(n)] = T_i$ and $k = 0{\cdot}95$ is chosen experimentally. Thus:

$$c_i(n + 1) = k\, c_i(n) + (1 - k)\, T_i$$

T_i (normalized within the interval [0, 1]) is the mean packet delay of the packet transmitted through the ith VC, during the time of INTERVAL arrivals at the destination node.

Measurement of VC-Delay:

Current mean packet delay of a VC can be measured at the VC's source by keeping track on the round trip packets delay (delay until a packet's ACK is received). An alternative approach is the measurement at the VC's destination. The delay of each packet is extracted by a control information attached on it.

Normalization:

In every LA the mean value of T_i's is kept. Any T_i is normalized after its division by the relevant maximum value (of T_i's) which is the double of the mean value of T_i's kept. Values greater than the relevant max are normalized to the value 1. The technique is depicted in Figure 6.

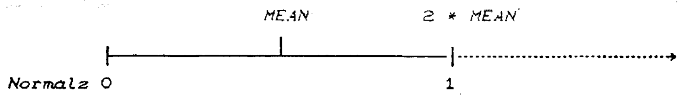

Fig. 6.

Remarks on the Function VC-Delay:

1. It has the properties (3) stated in Section 2.3. T_i is proportional to the probability p_i. Increase in p_i means increase in number of permits assigned to the ith VC which means increase of traffic through the VC and thus increase in T_i (by M/M/1 rules).
2. As T_i takes high values the corresponding penalty probability $c_i(n)$ increases in value and as a result, the LA reduces the probability of the ith VC-action (p_i). Consequently, the LA rewards (provides with more permits) the VC's with lower packet delays.

4.3 Updating of the Action Probabilities

Formulas (1) are used, with parameters $a = b = 0{\cdot}02$ (linear REWARD-PENALTY model). If the penalty probability of the chosen ith VC is lower than a THRESHOLD value, then $\beta(n) = 0$, else $\beta(n) = 1$.

5. SIMULATION RESULTS, COMPARISONS-CONCLUSIONS

5.1 Simulation for a 9-Node Network

Isarithmic-Learning Algorithm is compared with an isarithmic-Random Algorithm, which distributes the permits to the source nodes randomly and no probability vectors are recorded. In our simulation program, the network of Fig. 7 (9 nodes & & 4 VCs) was used first. In the case of Isarithmic-Learning, there are 2 LAs; one at node 4 with possible action {VC 1, VC 4} and the second at node 6 with possible action {VC 2, VC 3}.

For each Algorithm (RANDOM ISAR, LEARNING) simulation has run 10 times, with 35 000 EVENTS each time and different OfferedLoad each time. Final

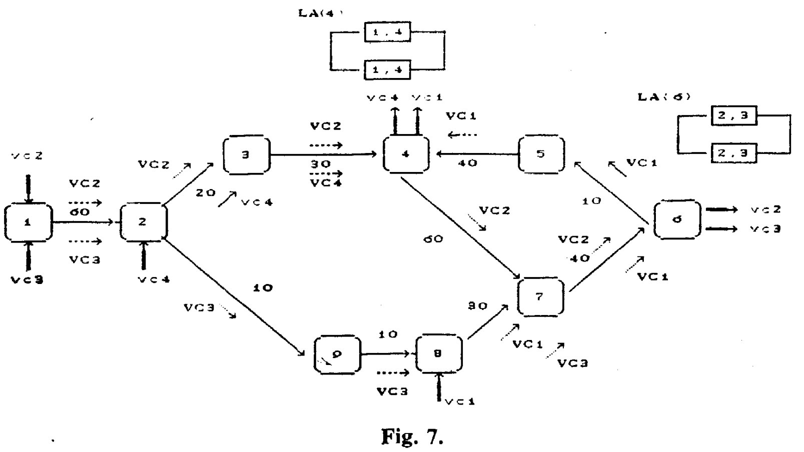

Fig. 7.

results have driven to the following graphs

throughput	$= f$ (OfferedLoad)	(Fig. 9)
Delay	$= f$ (OfferedLoad)	(Fig. 10)
Power	$= f$ (OfferedLoad)	(Fig. 11)

After observation of Simulation results and the corresponding graphs we came to the following conclusions. The Learning Algorithm gives (particularly at the high loads):

- Slightly higher Throughput than Random. We must note that both Algorithms succeed in Flow Controlling the network (the Throughput graphs show that the throughput climbs up to a limit imposed by the isarithmic flow control scheme adopted).
- A much better (= lower) mean packet delay than Random Algorithm. This implies that the LAs 'operated' as desired and the environment function T_i resulted in an improved operation of the total network, by choosing the actions with lower delay overhead.
- Improvement in power of the network, which is an obvious consequence of the above.

5.2 Simulation for a 4-Node Network

The network depicted in Figure 8 was also used for simulating the two Algorithms operation.

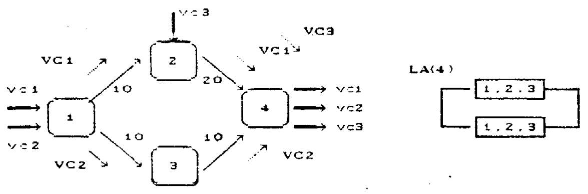

Fig. 8.

The corresponding graphs are shown as follows:

throughput	$= f$ (OfferedLoad)	(Fig. 12)
Delay	$= f$ (OfferedLoad)	(Fig. 13)
Power	$= f$ (OfferedLoad) = Throughput/delay	(Fig. 14)

The conclusion is that Learning Algorithm gives better results than Random, especially in the high loads.

It is worth to note that the results deviation between Learning & Random Algorithms is greater than the deviation in the previous network. This happens because the VCs differ more in traffic capability in the latter network, which results in a stronger variance in the mean packet delays of the 3 VCs.

5.3 Parameters in Learning Algorithm

The following parameters were used in the Isarithmic Learning Algorithm:
– $a = b = 0{\cdot}02$, $k = 0{\cdot}95$: [7].

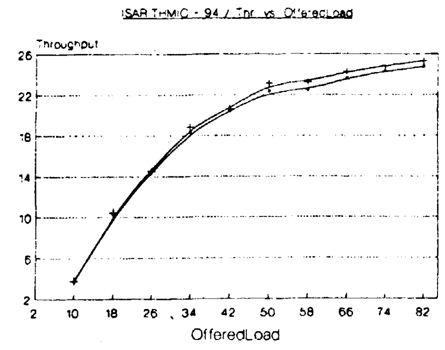

Fig. 9.

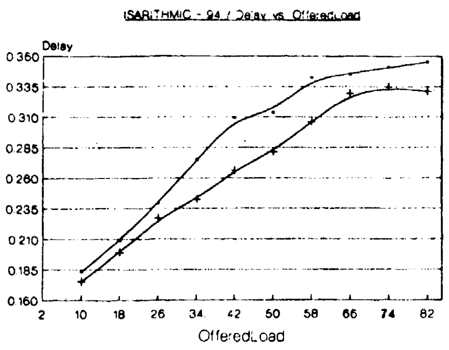

Fig. 10.

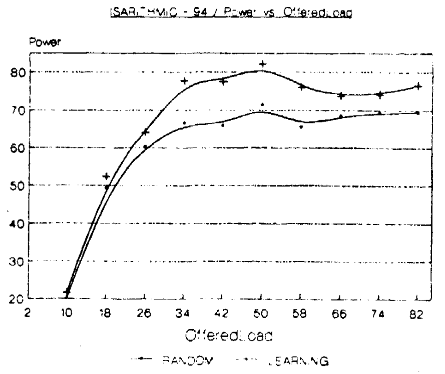

Fig. 11.

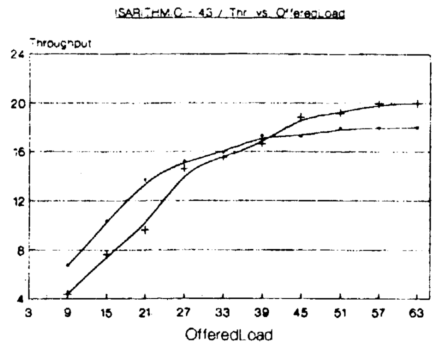

Fig. 12.

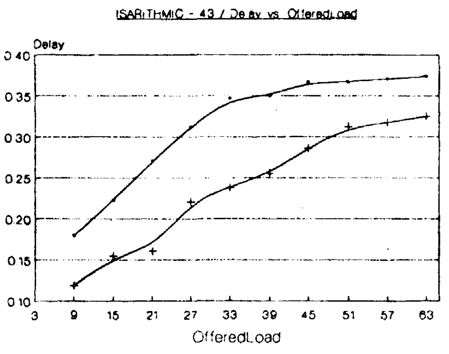

Fig. 13.

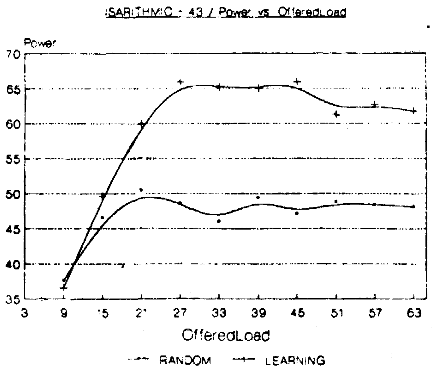

Fig. 14.

– THRESHOLD = 0·3: set to this value after long tests.
– INTERVAL = 4: this number should be close to parameter PERMITS VC.
– PERMITS VC = 5: chosen after tests. Isarithmic-Learning Algorithm gives better throughput for a choice of 6 permits/VC (than with 4, 5). This is reasonable since more permits through the whole network result in more allowable load in it. This notion is depicted in the following graphs:

Figure 7 network
Throughput $= f$ (OfferedLoad) (Fig. 15)
Figure 8 network
Throughput $= f$ (OfferedLoad) (Fig. 16).

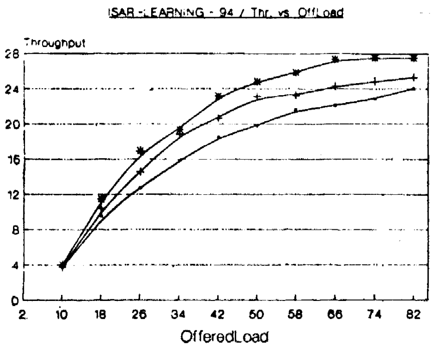

Fig. 15.

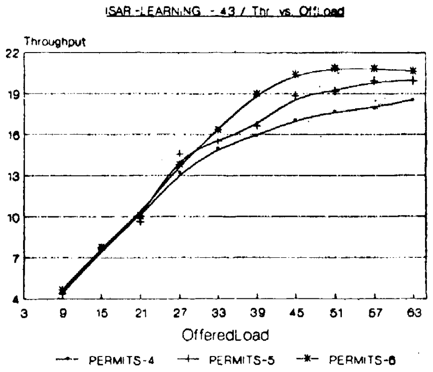

Fig. 16.

6. CONCLUSION

In this work, we have presented the "Isarithmic-Learning" algorithm, which uses Learning Automaton in order to solve the permit distribution problem. The simulation results indicate that the use of this algorithm gives a good performance for a packet-switched network.

The speed of Learning Automaton convergence and a better choice of the algorithm parameters are problems for further investigation. We hope that discretized learning algorithms will improve dramatically the performance of the FC scheme. We are currently working on this field.

(Received July 21, 1988.)

REFERENCES

[1] A. S. Tanenbaum: Computer Networks. Prentice-Hall, Englewood Cliffs, N. J. 1981.
[2] D. Bertsekas and R. Gallager: Data Networks. Prentice-Hall, Englewood Cliffs, N. J. 1987.
[3] A. V. Vasilakos and S. A. Koubias: On routing and performance comparison of techniques for packet-switched networks using learning automata. In: Proc. IEEE International Conference on Circuits and Systems, Espoo, Finland, June 1988.
[4] Narendra and M. Thathachar: Larrning automata: a survey. IEEE Trans. Systems Man Cybernet. *4* (1974), 323—334.

[5] G. Thaker and J. Cain: Interactions between routing and flow control algorithms. IEEE Trans. Comm. *34* (1986), 269—277.

[6] M. Gerla and L. Kleinrock: Flow control: a comparative survey. IEEE Trans. Comm. *28* (1981), 553—574.

[7] A. V. Vasilakos and S. A. Koubias: The use of learning algorithms in data network routing: a methodology. In: Proc. IFIP TC 6/TC 8 International Symposium, Network Information Processing Systems, Sofia, Bulgaria, May 1988.

[8] H. Kobayashi: Modelling and Analysis: An Introduction to System Performance Evaluation Methodology. Addison-Wesley, Reading, Mass. 1981.

Dr. A. V. Vasilakos, Department of Computer Engineering, University of Patras, 26500 Patras and Computer Technology Institute (CTI), Patras. Greece.

A. Haritsis, Department of Computer Science, Imperial College, London. England.

S. Batistatos, Department of Computer Engineering, University of Manchester, Manchester. England.